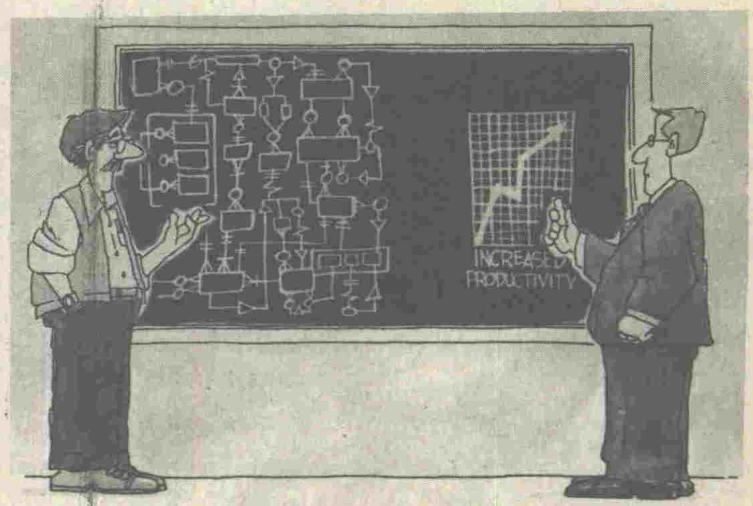
ฉัตรชัย ศิริรัมย์

เรียนรู้ภาษา โพสต์สคริปต์ ตอนที่ 1

ศึกษาเรื่องราวเกี่ยวกับ ประวัติความเป็นมา และคุณสมบัติต่าง ๆ ของภาษาโพสต์สคริปต์



ในปัจจุบันมีภาษาคอมพิวเตอร์มากมายหลายภาษา แต่ละภาษาก็มีลักษณะที่แตกต่างกันทางด้านความสามารถ ซึ่งอาจจะมุ่งเน้นไปด้านใดด้านหนึ่งโดยเฉพาะ เป็นการเพิ่มความสะดวกให้มนุษย์ได้ติดต่อสั่งงานคอมพิวเตอร์ให้เกิดประสิทธิภาพยิ่งขึ้นนั่นเอง แล้วแต่ว่าเราจะนำไปประยุกต์ใช้งานในด้านใด ซึ่งเราก็คงรู้จักภาษาต่าง ๆ อาทิ เช่น ภาษาซี ปาสคาล แอสเซมบลี เบสิก โคบอล RPG เป็นต้น ภาษาต่าง ๆ เหล่านี้คงเป็นที่รู้จักคุ้นเคยกันพอสมควร แต่เราน่าจะมาลองศึกษาภาษาใหม่กันดูบ้าง ซึ่งมีภาษาหนึ่งที่มีความน่าสนใจอยู่ไม่น้อยทีเดียว ภาษาที่ว่านี้ก็คือ ภาษา PostScript นั่นเอง

ภาษาโพสต์สคริปต์ได้ถูกคิดค้นโดยบริษัทอะโดบี้ซิสเต็มส์ (Adobe Systems Inc.) ซึ่งภาษาดังกล่าวให้วิธีการใหม่อันหน้าตื่นเต้นในการสร้างเอาต์พุตที่ยุ่งยากซับซ้อนได้ โดยจะเน้นหนักไปในเรื่องของการแสดงผลทางด้านกราฟิก และงานสิ่งพิมพ์โดยเฉพาะ อีกทั้งตัวของภาษานี้ได้ถูกกล่าวขวัญว่าเป็นภาษาหลักเพื่อการออกแบบโปรแกรมในการควบคุมเครื่องพิมพ์เป็นพิเศษ หรือที่เราจะเรียกว่า Page Description Language (PDL) โดยปัจจุบันนี้เทคโนโลยีของโพสต์สคริปต์เองได้รับการสนับสนุนจากบริษัทยักษ์ใหญ่มากมาย อาทิเช่น Apple Computer, NeXT, IBM, Microsoft, Sun Microsystems ฯลฯ ซึ่งเป็นการดีที่จะทำให้ภาษาโพสต์สคริปต์เป็นที่รู้จักแพร่หลายมากขึ้น เพราะด้วยประสิทธิภาพและความสามารถที่มันมีอยู่อาจจะทำให้ผู้อ่านหลงใหลจนอยากสัมผัสกับมันก็เป็นได้

ความเป็นมาของโพสต์สคริปต์

ภาษาโพสต์สคริปต์นั้นได้กำเนิดขึ้นในปี ค.ศ. 1985 โดยบริษัทอะโดบี้ซิสเต็มส์ ได้ร่วมกับบริษัทแอปเปิ้ลสร้างเครื่องพิมพ์ Apple Laser Writer ซึ่งเป็นเครื่องพิมพ์ที่เอื้ออำนวยในการเขียนภาษาโพสต์สคริปต์ (เราจะเรียกเครื่องพิมพ์นั้นว่า PostScript Printer) นับได้ว่าเป็นอุปกรณ์ที่ได้รับความนิยมในการเขียนโปรแกรมภาษาโพสต์สคริปต์มากทีเดียว และบริษัทอะโดบี้ซิสเต็มส์ได้เรียกภาษาโพสต์สคริปต์รุ่นนี้ว่า PostScript Level 1 โดยในเวลาต่อมาก็ได้มีการประดิษฐ์คิดค้นเครื่องพิมพ์สีขึ้นประกอบกับตลาดการค้าของบริษัทแอปเปิ้ลในญี่ปุ่นได้เติบโตขึ้น ดังนั้นบริษัทอะโดบี้ซิสเต็มส์จึงได้มีการพัฒนาโดยได้มีการเพิ่มขีดความสามารถขึ้นในด้าน Color Models และ Composite Fonts ในขณะเดียวกันบริษัท NeXT ได้เข้ามาร่วมมือกับบริษัทอะโดบี้ซิสเต็มส์สร้างจอภาพที่สามารถแสดงผลจากการ

เขียนโปรแกรมภาษาโพสต์สคริปต์ได้ ซึ่งถือว่าเป็นต้นกำเนิดของ Display PostScript นั่นเอง และในปี ค.ศ. 1990 บริษัทอะโดบี้ซิสเต็มส์ได้มีการปรับปรุงภาษาโพสต์สคริปต์ครั้งใหญ่ โดยทำการรวมเอาความสามารถด้านต่าง ๆ ของภาษา PostScript Level 1, Display PostScript, Color Models, Composite Fonts ฯลฯ เข้าไว้ด้วยกัน เพื่อเป็นการเพิ่มประสิทธิภาพของภาษา แล้วเรียกภาษาโพสต์สคริปต์ดังกล่าวนี้ว่า PostScript Level 2 ซึ่งถือว่าเป็นรุ่นล่าสุดในปัจจุบันนี้ แต่ที่ผู้เขียนจะยกตัวอย่างโปรแกรมต่าง ๆ เพื่อประกอบคำอธิบายนั้นส่วนใหญ่จะเป็น Display PostScript แล้วหากจะถามว่า Display PostScript เป็น PostScript level ใด ก็คงต้องกล่าวว่า เป็น level ที่กึ่งกลางระหว่าง PostScript Level 1 กับ PostScript Level 2

เครื่องมือที่ใช้กับภาษาโพสต์สคริปต์

อุปกรณ์สำคัญที่สนับสนุนการเขียนโปรแกรมด้วยภาษาโพสต์สคริปต์นี้ ส่วนใหญ่จะเป็นเครื่องพิมพ์ที่มีคุณสมบัติเป็นโพสต์สคริปต์ ซึ่งถือว่าเป็นอุปกรณ์ที่มีความละเอียดในระดับปานกลาง เครื่องพิมพ์โพสต์สคริปต์ต่าง ๆ ได้แก่ Apple Laser Writer, Micro Laser, OKI OL830 ฯลฯ หากท่านผู้อ่านต้องการจะเลือกซื้อเครื่องพิมพ์เลเซอร์ที่เราจะสามารถใช้ภาษาโพสต์สคริปต์ด้วย ก็ควรมีการสอบถามกับผู้ขายเสียก่อนว่าเลเซอร์พรินเตอร์นั้นซัพพอร์ตการเขียนโปรแกรมภาษาโพสต์สคริปต์หรือไม่ ท่านผู้อ่านจะได้มีการทดลองเขียนโปรแกรมต่าง ๆ ของภาษาโพสต์สคริปต์กันดูบ้าง

นอกจากนี้เรายังสามารถเขียนโปรแกรมภาษาโพสต์สคริปต์ให้แสดงผลลัพธ์ทางจอภาพได้ (เราจะเรียกภาษาโพสต์สคริปต์นี้ว่า Display PostScript) โดยมันจะต้องอาศัยโปรแกรม interpreter เพื่อให้สามารถแสดงผลลัพธ์ทางจอภาพได้ ซึ่งที่ผู้เขียนเคยทดลองเขียนโปรแกรมโพสต์สคริปต์ในลักษณะดังกล่าว คือ ใช้โปรแกรม PageView (ถือว่าเป็นโปรแกรม interpreter) ซึ่งทำงานบนเครื่อง Sun SPARC station IPC ของบริษัท ซัน ไมโครซิสเต็มส์ และการเขียนโปรแกรมภาษาโพสต์สคริปต์นั้นโดยทั่วไปต้องอาศัยเท็กซ์เอดิเตอร์ทำการเขียนโปรแกรมที่เป็นเท็กซ์ไฟล์ธรรมดา แล้วจึงจะส่งเท็กซ์ไฟล์นั้นไปยังเครื่องพิมพ์หรืออุปกรณ์แสดงผล (หากเป็นเครื่องพิมพ์ เครื่องพิมพ์นั้นควรจะใช้โปรโตคอลแบบสื่อสารได้สองทาง) และที่สำคัญในการเขียนโปรแกรมที่เป็นเท็กซ์ไฟล์ของภาษาโพสต์สคริปต์นั้นในบรรทัดแรกจะต้องกำหนด %! ไว้ก่อนเสมอ เพื่อเป็นการบ่งบอกว่าเป็นเท็กซ์ไฟล์ของภาษาโพสต์สคริปต์ หรือเสมือนว่าเป็นการเริ่มต้นของโปรแกรมนั่นเอง

นอกจากอุปกรณ์ต่าง ๆ ดังที่ได้กล่าวมาแล้วข้างต้นนั้น ยังมีอุปกรณ์อีกชนิดหนึ่งที่สามารถจะซัพพอร์ตภาษาโพสต์สคริปต์ นั่นก็คือ Typesetter ซึ่งถือว่าเป็นอุปกรณ์ที่มีความละเอียดสูง แต่เนื่องด้วยตัวผู้เขียนเองไม่ใคร่จะคุ้นเคยกับอุปกรณ์ชนิดนี้เท่าใดนัก จึงจะขอกล่าวถึงแต่เพียงเล็กน้อย ซึ่งนับเป็นสิ่งที่โชคดีอย่างหนึ่งก็คือ ภาษาโพสต์สคริปต์จะไม่ขึ้นอยู่กับความละเอียดของอุปกรณ์ เราไม่ต้องมาพะวงว่าหากเราเขียนโปรแกรมภาษาโพสต์สคริปต์ใด ๆ ขึ้นมาในอุปกรณ์หนึ่ง ๆ ที่สามารถใช้ภาษาโพสต์สคริปต์แล้ว เรานำโปรแกรมนั้นมาใช้ในอุปกรณ์ทางโพสต์สคริปต์อื่น แล้วจะเกิดเออร์เรอร์หรือให้ผลลัพธ์ที่มีความผิดแปลกจากที่ได้ใช้ในอุปกรณ์เดิม

มุมมองของโพสต์สคริปต์

โพสต์สคริปต์จะมองรูปทรงทุกอย่างเป็นกราฟิก (รวมทั้งตัวอักษรด้วย) แต่จะแตกต่างจากภาษาอื่นตรงที่ไม่อยู่ในรูปของระบบบิตแม็พ กล่าวคือ ถ้าเราต้องการพิมพ์ภาพกราฟิกใด ๆ เราไม่จำเป็นต้องส่งบิตแม็พของภาพนั้นให้ไปปรากฏที่กระดาษที่ต้องการพิมพ์ เพียงแต่เราส่ง algorithm ไปให้เท่านั้น โดยหลักการดังกล่าวจึงทำให้เครื่องพิมพ์โพสต์สคริปต์จำเป็นต้องมีส่วนประมวลผลผลภาพ (Raster Image Processor หรือ RIP) ที่คอยทำการแปลภาษาในลักษณะของ interpreted (แปลทีละคำสั่ง) และจะบรรจุ PostScript Interpreter ส่วนอุปกรณ์แสดงผลก็จะต้องมีซอฟต์แวร์พิเศษที่ทำหน้าที่เป็น Rasterizer โดยที่ตัวแปลภาษานั้นจะทำการแปล algorithm ที่ส่งมาให้เป็นภาพเสมือนก่อน (virtual image) หลังจากนั้นจึงจะส่งภาพเสมือนนั้นไปพิมพ์โดยใช้ระบบบิตแม็พตามความละเอียดของการแสดงผลของอุปกรณ์ที่เราใช้อยู่โดยอัตโนมัติ และรูปทรงที่ปรากฏเป็นภาพเสมือนนั้นจะเกิดจากการกระทำของฟังก์ชันทางคณิตศาสตร์ (โดยใช้ Cubic Bezier) โดยจะเก็บข้อมูลของทิศทาง (vector) ของเส้นรอบรูป (outline) นั้นไว้ และจะมีการปรับภาพให้ดูคมชัดมากยิ่งขึ้นอีกด้วย

ดังนั้นข้อได้เปรียบของเทคโนโลยี

โพสต์สคริปต์นี้สามารถสรุปได้ 3 ประการ

ข้อได้เปรียบประการที่หนึ่ง คือ สามารถย่อหรือขยายขนาดของตัวอักษร (Scalable) ได้ หรือแม้แต่จะให้ตัวอักษรนั้นเอียงมากน้อยตามทิศทางที่ต้องการ โดยอาศัยฟอนต์เพียงฟอนต์เดียวและไม่สูญเสียความคมชัด (ซึ่งเป็นฟอนต์ที่ใช้เทคโนโลยีของเอาต์ไลน์ฟอนต์) ซึ่งผิดกับฟอนต์แบบบิตแม็พ เพราะฟอนต์ของบิตแม็พจะมีจำนวนจุดคงที่เหมือนกับตรายาง ทุก ๆ อักขระในแต่ละขนาดที่แตกต่างกัน ไม่ว่าจะเป็นตัวเข้ม ตัวเอียง ตัวหนา ฯลฯ จะต้องมีบิตแม็พเป็นของมันเอง ซึ่งทำให้เสียเวลาในการออกแบบและเปลืองเนื้อที่ในการเก็บและการจัดการไฟล์

ข้อได้เปรียบประการที่สองก็คือ การเขียนโปรแกรมจะไม่ขึ้นอยู่กับความละเอียดของอุปกรณ์ (Resolution Independent) ไม่ว่าจะเป็นอุปกรณ์ที่มีความละเอียดแตกต่างกัน หลักการของการเขียนโปรแกรม และ Syntax ต่าง ๆ ของภาษาโพสต์สคริปต์ก็ใช้ร่วมกันได้

และข้อได้เปรียบประการสุดท้าย มันจะไม่ขึ้นอยู่กับชนิดของอุปกรณ์ (Device Independent) คือ มีความคอมแพตทิเบิล ไม่ว่าอุปกรณ์ชนิดใดหากมี interpreter ของภาษาโพสต์สคริปต์เท่านั้น เราก็สามารถนำโปรแกรมของภาษาโพสต์สคริปต์นั้น ๆ มาใช้กับอุปกรณ์ดังกล่าวได้

หลักการของภาษา

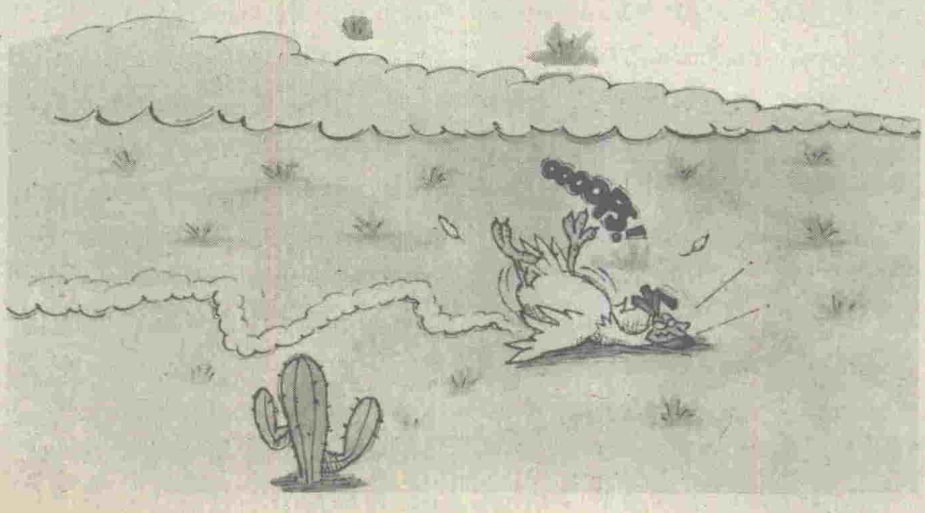
ภาษาโพสต์สคริปต์จัดได้ว่าเป็นภาษาที่ใช้สแต็กเป็นพื้นฐานสำคัญในการเก็บข้อมูล การผ่านค่าและการเก็บโค้ดที่สามารถประมวลผลได้ โครงสร้างของภาษาโพสต์สคริปต์จะแตกต่างจากภาษาอื่น ๆ โดยแต่ละประโยคคำสั่งจะถูกพิจารณาเป็นแบบถอยหลัง หรือที่เรียกว่า backward (ซึ่งก็เป็นวิธีการเดียวกันกับ post-fix) เช่น เราจะทำการบวกค่า 1 กับ 2 เข้าด้วยกัน หลักการของ backward จะต้องเขียนตามลักษณะดังนี้

```
2 1 add
```

จากตัวอย่าง ค่า 1 จะถูกกำหนดให้เป็นค่าแรก (ตัวตั้ง) และค่า 2 จะเป็นค่าที่สอง ซึ่งทั้งสองค่าจะถูกผลักลงสแต็กก่อน แล้วจึงนำค่าทั้งสองไปให้แก่ add operator โดยจะถือว่าค่าของทั้งสองเป็น operand เมื่อกระทำการบวกค่าเสร็จแล้ว ค่าทั้งสองจะถูก pop ออกจากสแต็กโดยอัตโนมัติ หลังจากนั้น add operator จะนำค่าที่เป็นผลลัพธ์ผลักลงสแต็กแทน (ในที่นี้ 3 คือผลลัพธ์ซึ่งก็จะถูกผลักลงสู่สแต็ก) ด้วยเหตุนี้เอง ภาษาโพสต์สคริปต์จึงไม่จำเป็นต้องใช้ second storage ที่มีขนาดใหญ่ในการเก็บข้อมูล ซึ่งข้อมูลและโค้ดต่าง ๆ จะถูกมองเป็นออปเจ็กต์ กล่าวคือ ถ้าเป็นออปเจ็กต์ของโค้ดจะสามารถประมวลผลได้ และออปเจ็กต์นั้นจะไม่ถูกแบ่งแยกออกเป็นโกลบอลและโลคอลอย่างในภาษาอื่น ๆ

สแต็กคือพื้นฐานสำคัญที่ต้องเข้าใจ

สแต็กเป็นสิ่งสำคัญที่สุดในภาษาโพสต์สคริปต์ โดยเป็นพื้นฐานที่ถูกใช้สำหรับการส่งผ่านอาร์กิวเมนต์ไปยังโอเปอเรเตอร์หรือโพรซีเดอร์ และยังเป็นที่ใช้สำหรับการส่งค่ากลับมายังสแต็กเพื่อนำไปใช้งานต่อไป ในภาษาโพสต์สคริปต์นั้นจะประกอบไปด้วยสแต็ก 4 ชนิด ก็คือ Operand stack, Execution stack, Dictionary stack และ Graphic state stack ซึ่งการทำงานของสแต็กต่างๆ จะเป็นลักษณะ last in first out (LIFO) และสแต็กที่นำมาใช้งานบ่อยที่สุดในการ execute ของโปรแกรมภาษาโพสต์สคริปต์ก็คือ operand stack ซึ่งถือว่าเป็นสแต็กหนึ่งที่มีความสำคัญมากในภาษาโพสต์สคริปต์ นอกจากจะใช้เป็นที่ส่งค่าอาร์กิวเมนต์ของคำสั่งต่าง ๆ แล้ว operand stack ยังเป็นที่สำหรับเก็บออปเจ็กต์ที่ต้องการไว้ชั่วคราว ทำให้เราสามารถควบคุมการส่งผ่านค่าต่าง ๆ ได้โดยตรง



ส่วนสแต็กที่ถูกใช้เพื่อช่วยในการค้นหาออปเจ็กต์ที่ตรงกับชื่อที่ต้องการและเก็บชื่อที่นิยามเอาไว้ (ชื่อของโพรซีเดอร์) นั่นก็คือ Directory stack ซึ่งตัวของ Dictionary stack เองจะแบ่งส่วนประกอบออกเป็น 2 ส่วน คือ userdict และ systemdict โดย systemdict นี้จะบรรจุชื่อและโพรซีเดอร์ที่เป็นโอเปอเรเตอร์ต่าง ๆ ของภาษาโพสต์สคริปต์มาตรฐาน อาทิเช่น add และ sub operator ส่วน userdict จะเป็นที่ที่ให้ผู้ใช้สร้างโพรซีเดอร์ใหม่เก็บไว้

เพื่อให้ผู้อ่านได้เข้าใจรายละเอียดมากขึ้น ลองมาพิจารณาจากตัวอย่างต่อไปนี้

```
/inch { 72 mul } def
% define inch procedure
```

inch เป็นชื่อโพรซีเดอร์ใหม่ที่ผู้ใช้สร้างขึ้น โดยจะมีเครื่องหมาย "/" นำหน้าชื่อโพรซีเดอร์เพื่อบอกประกาศให้ทราบว่า inch เป็นชื่อโพรซีเดอร์ และ 72 mul ที่อยู่ในวงเล็บปีกกาจะเป็นสิ่งที่ต้องดำเนินการเมื่อมีการเรียก procedure inch โดยโพรซีเดอร์นี้มันต้องการอาร์กิวเมนต์ที่เป็นตัวเลข 1 ตัว เพื่อนำมาคูณกับค่า 72 เมื่อเราจะจบการประกาศโพรซีเดอร์ หลังจากเครื่องหมายปีกกาปิดจะต้องตามด้วย def operator เพื่อเป็นการนำโพรซีเดอร์ใหม่ไปใส่ไว้ใน userdict หากเราต้องการใส่คอมเมนต์เพื่อเป็นการบอกหรือสื่อความหมายของส่วนโปรแกรมให้เราและผู้อื่นเข้าใจ ด้วยการใส่เครื่องหมายเปอร์เซ็นต์ "%" แล้วตามด้วยคำอธิบาย ตัว interpreter จะไม่ทำการ execute ใด ๆ สิ่งที่อยู่หลังเครื่องหมายเปอร์เซ็นต์ ซึ่งการเขียนคอมเมนต์ในโปรแกรมถือว่าเป็นหลักการเขียนโปรแกรมที่ดีที่โปรแกรมเมอร์ทั้งหลายควรจะกระทำ

หลังจากที่เรามีการประกาศ procedure inch แล้ว เราก็สามารถนำโพรซีเดอร์นี้ไปใช้งานได้ดังตัวอย่างที่จะแสดงดังนี้

```
1 inch 2 inch moveto
% move current point
```

ตัว Interpreter จะทำการค้นหาชื่อ inch ใน userdict ซึ่งเป็นชื่อของโพรซีเดอร์ และจะมีการ execute ตามที่ procedure inch ได้ถูกนิยามเอาไว้ ซึ่งก็มีผลเช่นเดียวกับการเขียนดังนี้

```
1 72 mul 2 72 mul move to
% move current point
```

สำหรับ Execution stack นั้นจะทำหน้าที่ในการเรียกส่วนที่ถูก execute ได้ อาทิเช่น ตัวโพรซีเดอร์ ไฟล์ออปเจ็กต์ และออปเจ็กต์อื่น ๆ ที่สามารถ execute ได้ และสแต็กชนิดสุดท้าย คือ Graphics State stack จะใช้ในการเก็บและคืนสภาวะกราฟิกโดยใช้ gsave และ grestore operator ตามลำดับ ซึ่งจะขออธิบายการใช้ gsave และ grestore operator ประกอบในตัวอย่างโปรแกรมที่ 2 ในหัวข้อเรื่อง Output page และระบบโคออร์ดิเนตต่อไป

ในการจัดการ operand stack เราลองมาดูกันบ้างว่าโอเปอเรเตอร์ต่าง ๆ ต้องการอาร์กิวเมนต์อะไรบ้าง และการใช้โอเปอเรเตอร์นั้นให้ผลลัพธ์เป็นเช่นไรตามที่แสดงในตารางที่ 1

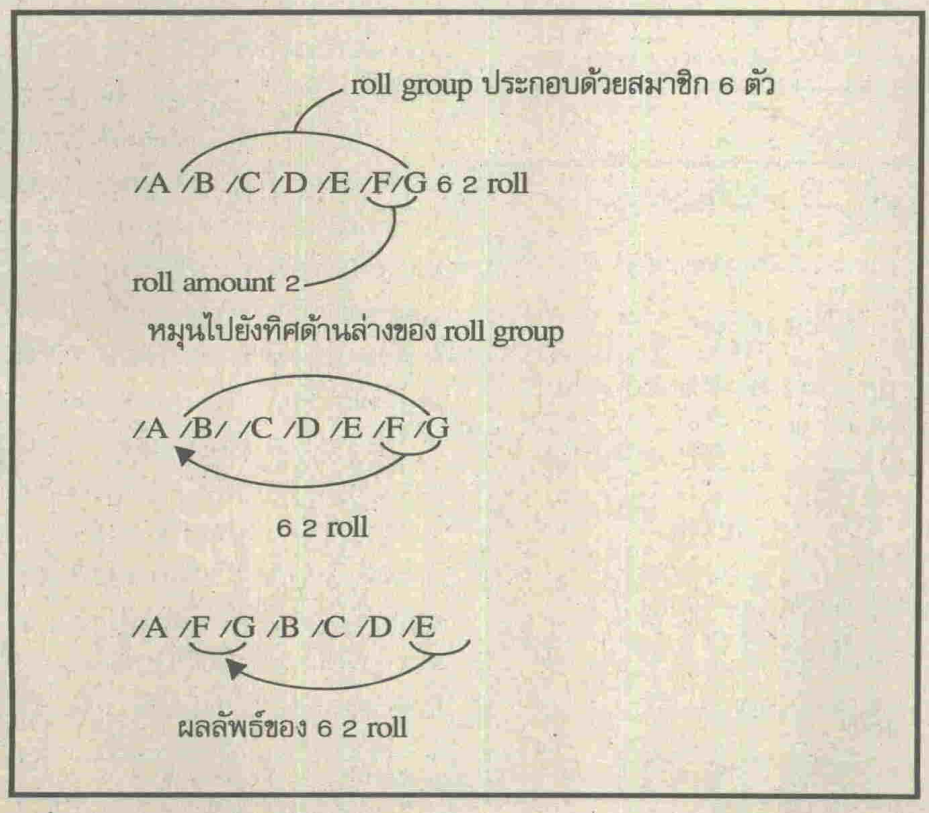
โอเปอเรเตอร์ต่าง ๆ ตามตารางที่ 1 เราจะอธิบายถึงโอเปอเรเตอร์ที่สำคัญ ๆ ในการจัดการ operand stack เท่านั้น คือ dub, exch, index และ roll การใช้ dup operator ใช้เพื่อจำลองออปเจ็กต์

ที่อยู่ตำแหน่งบนสุดของสแต็ก ซึ่ง dup operator เป็นโอเปอเรเตอร์ที่ค่อนข้างง่ายต่อการเข้าใจ แต่ถ้าหากเรารู้จักใช้มันก็จะเกิดประสิทธิภาพในการใช้งานได้ดีทีเดียว ส่วน index operator เป็นโอเปอเรเตอร์ที่มีลักษณะคล้ายคลึงกับ dup operator หากต่างกันตรงแต่เพียงว่า dup operator จะทำการจำลองออปเจ็กต์ที่อยู่บนสุดของสแต็กเท่านั้น แต่ index operator จะทำการจำลองส่วนใด ณ ตำแหน่งหนึ่งตำแหน่งใดของสแต็กก็ได้

อีกโอเปอเรเตอร์ที่ควรจะรู้จักคือ exch operator ซึ่งเป็นโอเปอเรเตอร์ที่ใช้สำหรับเปลี่ยนลำดับหรือสลับตำแหน่งของ 2 ออปเจ็กต์ที่อยู่บนสุดของสแต็ก เพื่อทำให้ลำดับในการทำงานของโอเปอเรเตอร์เป็นไปอย่างถูกต้อง ส่วนการใช้ roll operator นั้นค่อนข้างจะซับซ้อน แต่ก็เป็นโอเปอเรเตอร์หนึ่งที่มีประโยชน์มากสำหรับสแต็ก โดยโอเปอเรเตอร์นี้ต้องการอาร์กิวเมนต์ 2 ตัว ซึ่งตัวแรกจะเป็นการแสดงจำนวนกลุ่มของ element ที่จะทำการหมุน (เรียกว่า roll group) ส่วนอาร์กิวเมนต์ตัวที่สอง

ตารางที่ 1 โอเปอเรเตอร์ที่จัดการโอเปอแรนด์สแต็ก

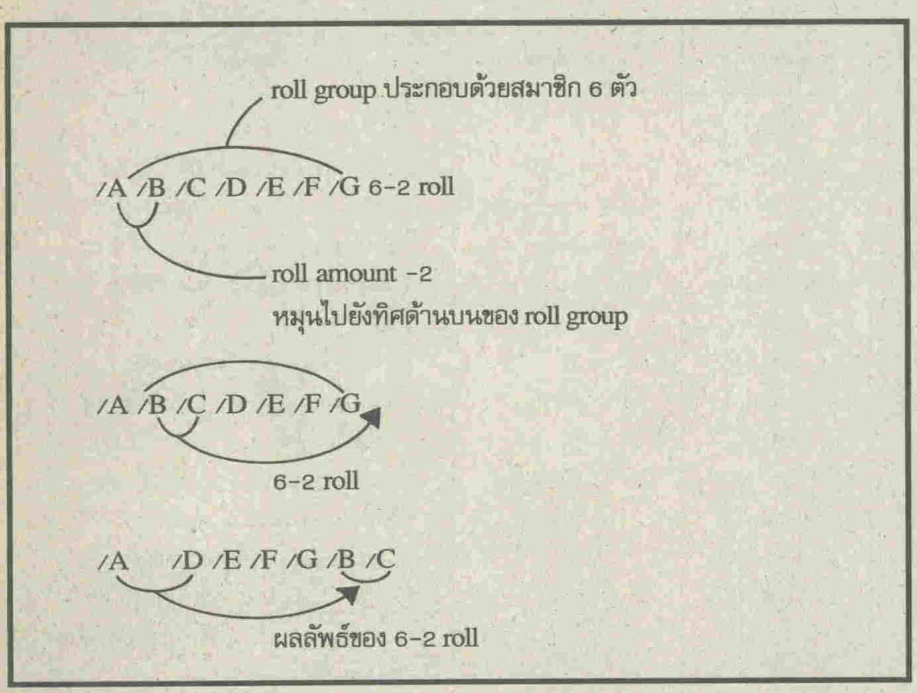
Arguments	Operator	ผลลัพธ์	หมายเหตุ
I any1...anyn	clear		ทิ้ง element ทั้งหมดในสแต็ก
mark obj1...objn	cleartomark		ทิ้งทุก ๆ element จนถึง mark
A B C 3	copy	A B C A B C	ก๊อบปี้ 3 element ที่อยู่บนสุด
I any1...anyn	count	any1...anyn n	นับจำนวน element บนสแต็ก
mark any__objects	counttomark	mark any n	นับจำนวน element ลงมายังที่ mark เอาไว้
any	dup	any any	จำลอง element ที่อยู่บนสุด
any1 any2	exch	any2 any1	สับเปลี่ยน 2 element ที่อยู่บนสุด
D E F G 2	index	D E F G E	จำลอง element ที่ 2 จากบนสุด (element บนสุดเป็นลำดับที่ 0)
mark	mark		push mark ลงบน stack
any	pop		pop element ที่อยู่บนสุดทิ้ง
a b c d e f 3-2	roll	a b c f d e	นำเอา 3 element บนสุดหมุนไปทางซ้าย 2 ครั้ง (ถ้าค่าแรกเป็นบวกจะหมุนไปทางขวา)



รูปที่ 1 การหมุนแบบ Positive

เป็นตัวบ่งชี้ทิศทาง และจำนวนครั้งของการหมุน (เรียกว่า roll amount) ซึ่งถ้าค่าของ roll amount เป็นค่าติดลบ เราจะเรียกว่าเป็นการหมุนแบบ negative กล่าวคือ จะหมุนในทิศทางจากด้านล่างขึ้นไปด้านบนของสแต็ก (Bottom ---> top) ตามจำนวนสมาชิกที่ระบุไว้ใน roll group และในทางกลับกันถ้า roll amount เป็นค่าบวกก็จะหมุนแบบ positive คือเป็นการหมุนในด้านตรงกันข้ามกับการหมุนแบบ negative นั่นเอง ดังจะได้แสดงดังรูปที่ 1 และรูปที่ 2

สิ่งสำคัญอีกประการหนึ่งในการทำความเข้าใจเกี่ยวกับสแต็กนอกเหนือจากโอเปอเรเตอร์ต่าง ๆ ที่ใช้สำหรับจัดการกับสแต็กก็คือ การจัดกลุ่มโอเปอแรนด์และโอเปอเรเตอร์ให้เหมาะสมเพื่อป้องกัน error ที่อาจจะเกิดขึ้นได้จากการวางตำแหน่งของโอเปอแรนด์ที่ผิดพลาดนั่นเอง ดังในตัวอย่างที่ 1 เราจะกล่าวถึงการค้นหา font Times-Roman ด้วย findfont operator ที่



รูปที่ 2 การหมุนแบบ Negative

ตัวอย่างที่ 1 การใช้ findfont Operator ค้นหาฟอนต์จากดิคชันนารี

```
/Times-Roman findfont (Testing) exch 12 scalefont setfont show
```

```
/Times-Roman findfont (Testion) exch 12 scalefont setfont show
/Times-Roman findfont (Testing) exch 12 scalefont setfont show
(Testing) /Times-Roman findfont 12 scalefont shco
(Testing) /Times-Roman findfont 12 scalefont setfont show
(Testing) [          ] show
```

หมายเหตุ

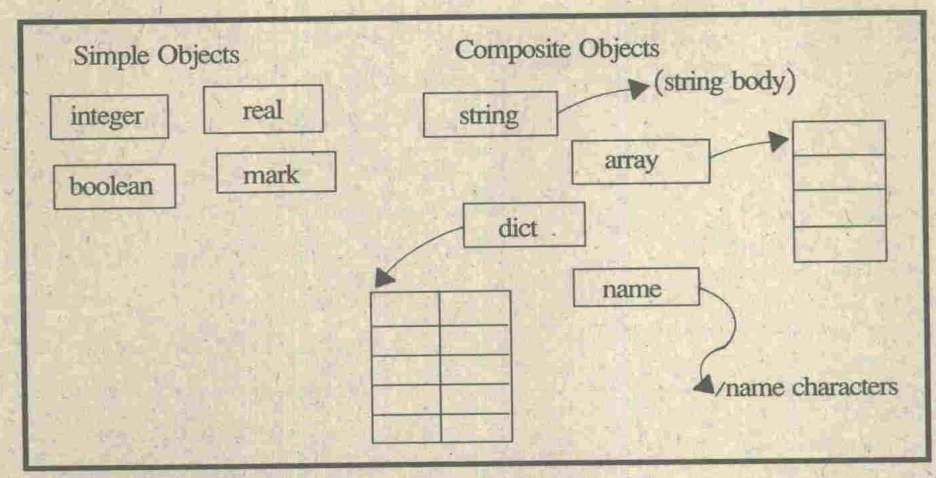
operators	การกระทำใน box จะผลิต object ใหม่ขึ้นมา
exch	กระทำตัวเข้มจะจัดเรียง operand ให้ใหม่
setfont	การกระทำตัวเอียงเข้มจะใช้ผลลัพธ์ที่เป็น object ใหม่เป็น operand
[ ]	box ว่างเปล่าเป็นตัวบ่งชี้ว่าค่าถูกนำออกจาก stack แล้ว

รูปที่ 3 การจัดกลุ่มของตัวอย่างที่ 1

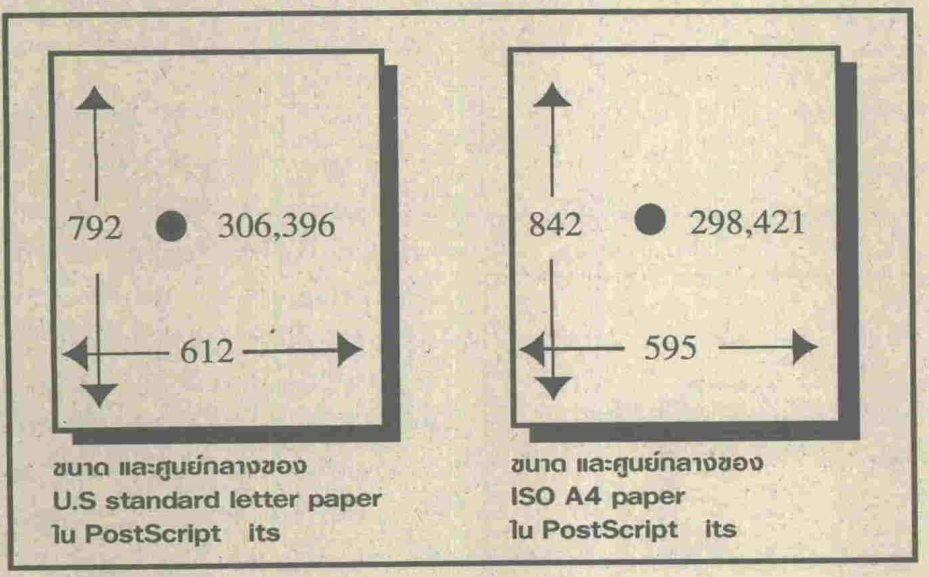
กำหนดใน dictionary stack โดยมีขนาดตัวอักษรเท่ากับ 2 ซึ่งเราจะระบุขนาดจาก scalefont operator และเมื่อค้นพบฟอนต์ใน dictionary stack แล้ว ก็จะใช้ setfont operator เป็นการบ่งบอกให้ทำการติดตั้งฟอนต์นั้นเป็นฟอนต์ปัจจุบัน แต่ถ้าหากการค้นหาไม่พบชื่อฟอนต์ดังกล่าว ฟอนต์ที่เป็นค่าดีฟอลต์ก็จะถูกนำมาติดตั้งแทน ส่วนข้อความ "Testing" จะถูกแสดงออกทาง output page ด้วย show operator และรูปที่ 3 เป็นการแสดงการจัดกลุ่มแต่ละขั้นตอน

ออปเจ็กต์โอเรียนเต็ด

ข้อมูลต่าง ๆ ที่เรานำมาใช้นั้นจะอยู่ของ single object ซึ่งเราสามารถแบ่งออปเจ็กต์ออกเป็น 3 ประเภท ประเภทแรกเรียกว่า simple objects เป็นออปเจ็กต์พื้นฐานที่ไม่ค่อยมีความซับซ้อนมากนัก อาทิเช่น integers, reals, masks และ booleans ส่วนประเภทที่สอง คือ composite objects ได้แก่ อะเรย์และสตริง ซึ่งขนาดของมันขึ้นอยู่กับการกำหนด หรือที่เราประกาศ และออปเจ็กต์ประเภทสุดท้ายเราเรียกว่า special objects เป็นออปเจ็กต์ที่มีลักษณะคล้ายคลึงกับ simple objects แต่โครงสร้างภายในบางอย่างเราไม่สามารถจะมองเห็นได้ special object จะได้แก่ save objects และ FontID objects สำหรับการเลือกใช้ออปเจ็กต์แต่ละประเภทก็ขึ้นอยู่กับจุดประสงค์ของการใช้งานว่าเป็นไปในลักษณะใด ซึ่งหลักการส่วนใหญ่แล้วการเลือกใช้ต่าง ๆ ก็เหมือนกับในภาษาอื่น ๆ ในรูปที่ 4 จะแสดงถึงชนิดของออปเจ็กต์ต่าง ๆ



รูปที่ 4 ชนิดของโพสต์สคริปต์ออปเจ็กต์ Simple and Composite



รูปที่ 5 แสดงถึงระบบโคออร์ดิเนตของกระดาษทั้งสอง

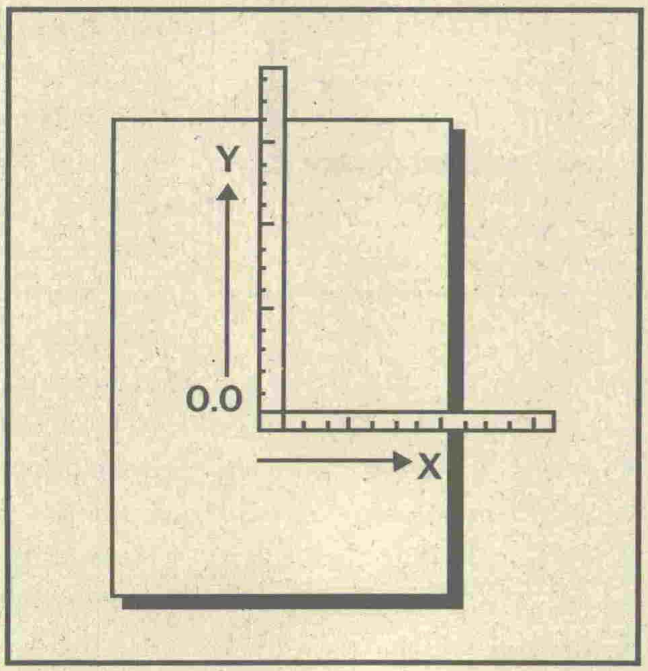
Output Page และระบบโคออร์ดิเนตเป็นอย่างไร

โพสต์สคริปต์ได้ชื่อว่าเป็น Page Description Language (PDL) ดังนั้นผลลัพธ์ที่ถูกแสดงออกมาจะเป็นในลักษณะของ output page แต่ละตำแหน่งบน output page (ทั้งบนหน้าจอและบนกระดาษ) เราจะเรียกหน่วยของและจุดนั้นว่าพิกเซล โดยพิกเซลหนึ่ง ๆ ของภาษาโพสต์สคริปต์จะมีค่าเท่ากับ 1/72 นิ้ว เสมอ (ใน 1 นิ้วจะมีอยู่ 72 จุด) และระบบโคออร์ดิเนตจะมีจุดมุมล่างซ้ายเป็นจุด origin (คือจุด 0,0) ส่วนความกว้างแกน x และความยาวแกน y จะขึ้นอยู่กับขนาดของ output page ด้วย ซึ่งขนาดของเอาต์พุตจะมีมาตรฐานต่าง ๆ อาทิเช่น U.S. standard letter paper จะมีความกว้างแกน x เท่ากับ 612 หน่วย (หรือเท่ากับ 8.5 นิ้ว) และความยาวแกน y เท่ากับ 792 หน่วย (หรือเท่ากับ 1 นิ้ว) ส่วน ISO A4 paper มีความกว้างแกน x เท่ากับ 595 หน่วย (หรือประมาณ 8.3 นิ้ว) และความยาวแกน y 842 หน่วย (หรือประมาณ 11.8 นิ้ว) ดังที่จะได้แสดงระบบโคออร์ดิเนตในรูปที่ 5 ซึ่งภาษาโพสต์สคริปต์จะไม่ขึ้นอยู่กับความละเอียดของอุปกรณ์ดังที่ได้กล่าวมาแล้วในช่วงแรก และระบบโคออร์ดิเนตที่ใช้ก็เสมือนว่ามีแต่เฉพาะ quadrant ที่ 1 เท่านั้น (มีแต่ค่าบวกซึ่งง่ายต่อการใช้) โดยในภาษาโพสต์สคริปต์นั้นจะมี current point เริ่มต้นที่จุด origin เสมอ แต่เราก็ยังสามารถย้าย current point และจุด origin ได้ด้วย moveto และ translate operator ดังตัวอย่างที่ 2 ซึ่งเป็นตัวอย่างแสดงถึงการใช้โอเปอเรชันทั้งสอง ส่วนผลลัพธ์จากการรันโปรแกรมในตัวอย่างที่ 2 จะแสดงไว้ในรูปที่ 8

จากตัวอย่างที่ 2 เราทำการเปลี่ยนจุด origin จากจุด (0,0) ไปยังกลางหน้ากระดาษ คือ ณ จุด (306,396) ด้วย translate operator (output page ที่ผู้เขียนใช้เป็น U.S. standard letter paper) และวาดวงกลมที่จุด origin ใหม่ที่จุด (0,0) (ซึ่งก็คือตำแหน่ง (306,396) เดิมนั่นเอง) โดยมีค่ารัศมีเท่ากับ 270 มุมเริ่มต้นที่ 0 องศา สิ้นสุดที่ 360 องศา เพื่อให้เกิดเป็นรูปวงกลม โดยใช้ arc operator และรูปที่ 7 เป็นการแสดงรายละเอียดอาร์กิวเมนต์ต่าง ๆ ของ arc operator เมื่อเสร็จสิ้นการวาดวงกลมแล้วนั้นถือว่าเราได้สร้างพาธใหม่ขึ้น ดังนั้นเราจึงต้องใช้ close-path operator ทำการปิดพาธนั้นเพื่อที่จะสามารถวาดวงกลมได้ หรือตีเส้นรอบวงนั่นเอง การจะเติมหรือระบายสี

ตัวอย่างที่ 2 การใช้ moveto และ translate operator

```
%IPS-Adobe
306 396 translate                % origin to center of page
0 0 270 0 360 arc                % draw a circle
closepath                        % finish the shape
gsave                            % remember graphics state
        0.5 setgray              % medium gray shade
        fill                     % fill circle
grestore                         % restore graphics state
72 setlinewidth                  % set a fat line
stroke                           % stroke the path
/Palatio-Bold findfont           % find a font
360 scalefont                    % make letters large
setfont                          % set current font
96 275 moveto                    % position current point
(PS) false charpath              % get character path
gsave                            % remember graphics state
        18 setlinewidth          % fat line width
        stroke                   % paint outline of characters
grestore                         % restore graphics state
1.0 setgray                      % white color
fill                             % fill character outlies
showpage                         % display page
```

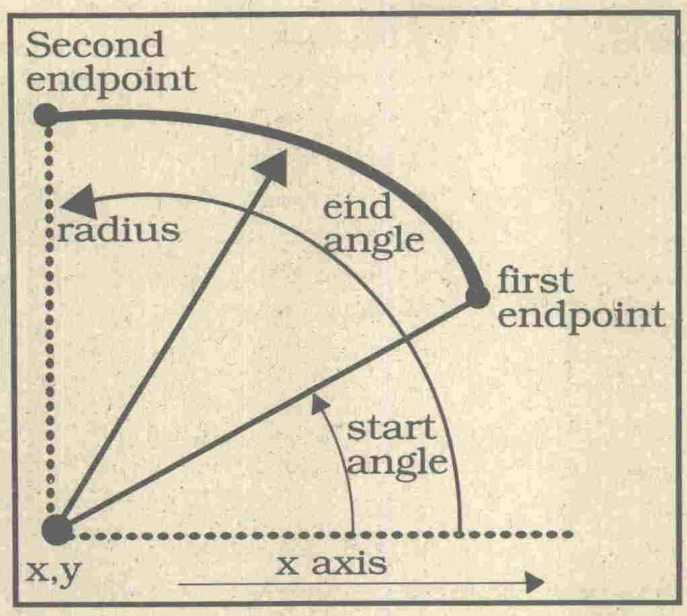


รูปที่ 6
การ translate ตำแหน่ง origin ไปยังระบบโคออร์ดิเนตของผู้ใช้

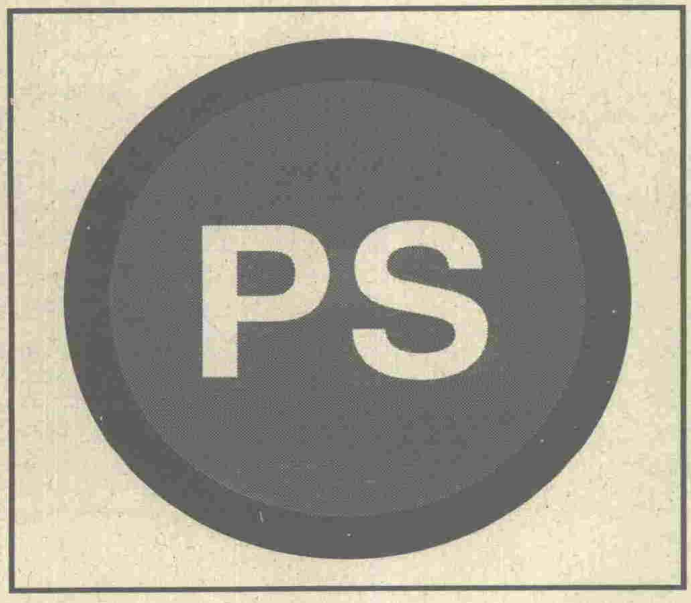
เราจะใช้ fill operator ซึ่งถ้าเราต้องการเปลี่ยนเชดสีใหม่จากค่าดีฟอลต์ที่มันเป็นอยู่ เราก็สามารถใช้ setgray operator โดยค่าอาร์กิวเมนต์จะเป็นค่าทศนิยมอยู่ระหว่าง 0 ถึง 1 (0 ถือเป็นสีที่เข้มที่สุด (ดำ) และ 1 เป็นสีที่อ่อนที่สุด (ขาว))

จากตัวอย่างแสดงว่าพื้นที่ภายในวงกลมด้านใน เราต้องการระบายสีเทาในระดับความเข้มปานกลาง เราจึงใช้คำสั่งว่า 0.5 setgray การเปลี่ยนเชดสีนั้นจะทำให้สภาวะกราฟิกปัจจุบันนั้นเปลี่ยนไป ดังนั้นหากว่าเราต้องการเก็บสภาวะกราฟิกไว้ ก็ต้องใช้ gsave operator และจะใช้ grestore operator เพื่อทำการคืนสภาวะกราฟิกให้กลับสู่สถานะเดิมก่อนที่มีการใช้ gsave operator หลังจากนั้นจะมีการตีเส้นทึบเพื่อเน้นภาพวงกลมให้มองเห็นได้ชัดเจนยิ่งขึ้น ด้วยการกำหนดเส้นหนา ๆ ให้มีขนาด 1 นิ้วด้วยคำสั่ง 72 setlinewidth แล้วให้เส้นนั้นถูกตีทึบลงไปด้วยคำสั่ง stroke

ต่อจากนั้นในการแสดงผลตัวอักษรเราสามารถค้นหาฟอนต์ที่ต้องการได้ใน dictionary stack ด้วย findfont operator โดยให้ชื่อฟอนต์ที่ต้องการใช้เป็นอาร์กิวเมนต์ของ findfont และการกำหนดขนาดของตัวอักษร เราจะใช้ scalefont operator ซึ่งก็แล้วแต่ว่าเราต้องการขนาดตัวอักษรเท่าใด ก็ใส่ค่าที่ต้องการลงในอาร์กิวเมนต์ อาทิเช่น เราต้องการขนาดตัวอักษร 5 นิ้ว ก็เขียนว่า 360 scalefont (1 นิ้ว = 72 หน่วย) สุดท้ายก็อย่าลืมใช้ setfont operator ด้วยนะครับ เพื่อเป็นการเปลี่ยนฟอนต์ที่ใช้อยู่ในปัจจุบัน และถ้าเราต้องการพาธที่เป็นรูปอักษร เราก็สามารถใช้ charpath operator โดยอาร์กิวเมนต์ของโอเปอเรเตอร์นี้ต้องการอาร์กิวเมนต์ 2 ตัว คือ ชุดอักขระที่ต้องการจะสร้างพาธ และค่าตรรกหรือค่า boolean (true,false) ซึ่งใช้สำหรับการ stroke path ส่วน true ใช้สำหรับการ fill path นั้น และคำสั่งสำคัญที่จะขาดเสียไม่ได้เลยก็คือ showpage ปิดท้ายโปรแกรม เพื่อให้ได้เอาต์พุตแสดงผลออกมาดังรูปที่ 8



รูปที่ 7
แสดงถึง parameters
ของ arc operator



รูปที่ 8
output page
ของตัวอย่างที่ 2

ประโยคเงื่อนไข

ในประโยคเงื่อนไขของโพสต์-สคริปต์นั้นจะต้องใช้โอเปอเรเตอร์ที่รีเทอร์นค่า boolean มาให้ และจะทำคำสั่งที่อยู่ในรูปของโพรซีเดอร์ หลังจากการตรวจสอบเงื่อนไขที่เป็นได้ ประโยคเงื่อนไขของโพสต์สคริปต์จะมีอยู่ 2 ลักษณะ คือ

```
- conditional operator {instruction,...} if
- conditional operator {instruction,...}
    {instruction,...} ifelse
```

จากประโยคเงื่อนไขข้างต้นจะเห็นได้ว่าภาษาโพสต์สคริปต์จะสนับสนุนภาษาที่เป็นแบบโครงสร้าง ดังนั้นจึงไม่มี goto operator และก็ไม่มีการใช้ case operator อย่างในภาษาอื่น ๆ อีกด้วย ส่วนตัวอย่าง conditional operator ที่ใช้บ่อยจะได้แสดงให้เห็นดังตารางที่ 2

เราลองมาดูตัวอย่างง่าย ๆ ของการใช้โอเปอเรเตอร์ในการเปรียบเทียบและค่า boolean ที่ได้

```
%comparing reals with integers:
103.3 103 eq        %true
103.0 103 ge        %true
103.0 103 gt        % false
```

การทำซ้ำในลักษณะเป็น Loop

โปรแกรมภาษาโพสต์สคริปต์ก็เช่นเดียวกันกับการเขียนโปรแกรมภาษาโครงสร้างอื่น ๆ ที่มักจะเขียนโปรแกรมในลักษณะมีการทำซ้ำ การใช้คำสั่ง loop ในภาษาโพสต์สคริปต์นั้นมี looping operator ต่าง ๆ ดังได้แสดงในตารางที่ 3

ถ้าเราต้องการ execute ลำดับคำสั่งเป็นจำนวนครั้งที่แน่นอน ควรจะเลือก repeat operator หรือ for operator เราจะใช้ repeat ถ้าไม่ต้องการใช้ loop index ภายใต้ตัวลูป และถ้าเราต้องการจะ execute loop จนกระทั่งคุณพบเงื่อนไขที่ตรวจสอบภายในก็ควรใช้ loop operator และจะใช้ exit เพื่อจะออกจากลูปในทันที ถ้าเราต้องการ execute คำสั่งบนทุก ๆ element ของอะเรย์และสตริง หรือดิคชันนารี คุณก็สามารถใช้ forall operator ตัวอย่างที่ 3 เป็นการใช้ลูปแบบ repeat ในการวาดรูปแปดเหลี่ยมแบบง่าย ๆ

จากตัวอย่าง 3 เรากำหนดเส้นในการวาดรูปแปดเหลี่ยมนี้ให้มีขนาดเท่ากับ 4 หน่วย ซึ่งเริ่มต้นจุดโคออร์-ดิเนตที่ (300,200) แล้วใช้ repeat loop เพื่อให้มีการดำเนินการทำซ้ำ (โดย

Arguments	Operator	ผลลัพธ์	หมายเหตุ
any1 any2	eq	bool	ตรวจสอบการเท่ากัน
any1 any2	ne	bool	ตรวจสอบการไม่เท่ากัน
obj1 obj2	ge	bool	ตรวจสอบการมากกว่าหรือเท่ากับ
obj1 obj2	gt	bool	ตรวจสอบการมากกว่า
obj1 obj2	le	bool	ตรวจสอบการน้อยกว่าหรือเท่ากับ
obj1 obj2	lt	bool	ตรวจสอบการน้อยกว่า
bool1 bool2	and	bool3	ตรรก and
bool1	not	bool2	ตรรก not
bool1 bool2	or	bool3	ตรรก inclusive or
bool1 bool2	xor	bool3	ตรรก exclusive or
	true	true	ใส่ค่าตรรก true ลงสแต็ก
	false	false	ใส่ค่าตรรก false ลงสแต็ก

ตารางที่ 2
ตัวอย่างโคออร์ดิเนต
โอเปอเรเตอร์

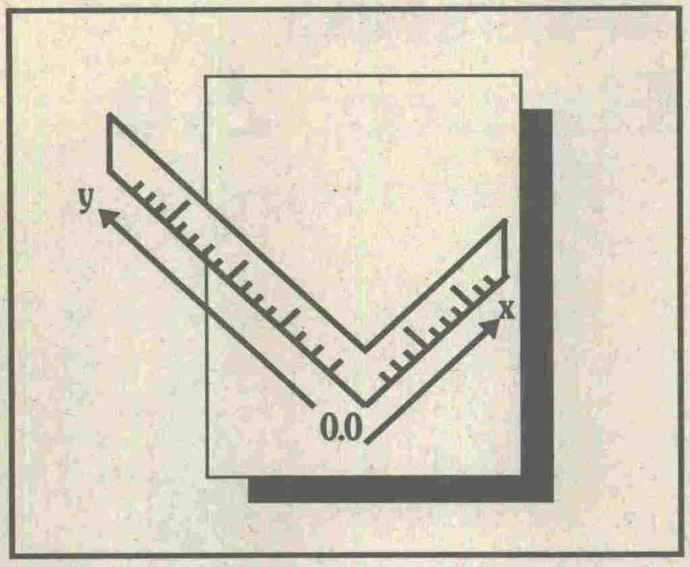
ตารางที่ 3 Looping Operators

Arguments	Operator	ผลลัพธ์	หมายเหตุ
—	exit	—	กระโดดออกจากลูปโดยทันที
pattern proc buffer	filenameforall		execute proc สำหรับแต่ละไฟล์ที่พบตาม pattern string
start indexend proc	for	—	execute proc หลาย ๆ ครั้ง ตาม (end-start) /เพิ่มทีละ index
composite_obj proc	forall	—	execute proc สำหรับแต่ละ element ของ composite_obj
proc	loop	—	execute proc ที่นิยามไว้
count proc	repeat		execute proc นั้นตามจำนวน count

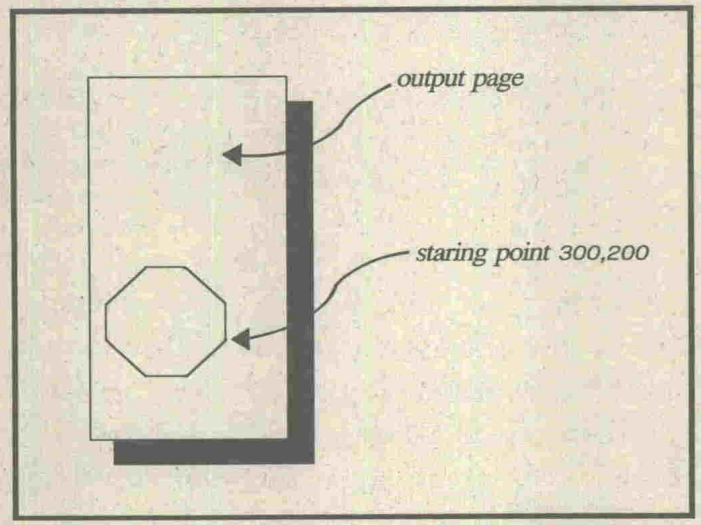
ตัวอย่างที่ 3 การวาดรูปแปดเหลี่ยม

```
%IPS-Adobe
4 setlinewidth                  % fat line width
300 200 moveto                  % position current point
7 { % repeat                    % repeat 7 times
      0 100 rlineto             % line to coordinate x and y+100
      45 rotate                 % rotate 45 องศา
} bind repeat                   % repeat loop
closepath                       % finish the shape
stroke                          % paint outline of characters
showpage                        % display page
```

กำหนดจำนวนครั้งที่จะกระทำซ้ำก่อนเข้าลูป) ภายในลูปครั้งแรกจะลากเส้นไปยังจุด (300,300) ด้วยคำสั่ง 0 100 rlineto คำสั่ง rlineto นั้น จะทำให้จุดใหม่ที่เกิดขึ้นเป็นจุดที่เกิดจากผลบวกของจุดโคออร์ดิเนตเดิม กับจุดใหม่ของทั้งแกน X และแกน Y จากตัวอย่างจุดเดิมก่อนเข้าลูปคือ (300,200) พอเข้าลูปใช้คำสั่ง 0 100 rlineto เพราะฉะนั้นจุดใหม่ที่ต้องลากเส้นตรงไปก็คือ (300,



รูปที่ 9
ผลของการ rotate
แกนไป 45 องศา
จากจุด origin



รูปที่ 10
เอาต์พุตของ
ตัวอย่างที่ 3

300) ซึ่งเกิดจาก (300+0,200+100) หลังจากนั้นจะทำการหมุนแกนของระบบโคออร์ดิเนตเป็นมุม 45 องศาด้วย rotate operator (หมุนทวนเข็มนาฬิกาจากจุด origin) ดังได้แสดงผลของการ rotate ดังรูปที่ 9

การวาดเส้นครั้งต่อ ๆ ไปก็จะดำเนินการจนครบจำนวนลูป แล้วใช้ closepath operator เพื่อปิดเส้นทางให้เกิดเป็นเส้นที่ต้องการ และใช้ stroke เพื่อตีเส้นทึบ และปิดท้ายด้วยคำสั่ง showpage เสมอ จะได้ผลลัพธ์รูปแปดเหลี่ยม ตามรูปที่ 10

รวมคำสั่งให้เป็นโพรซีเดอร์

ส่วนใหญ่แล้วภาษาโพสต์สคริปต์จะถูกเขียนให้อยู่ในรูปของ procedural เพื่อเป็นการรวบรวมคำสั่งต่าง ๆ ให้มีการทำงานที่มีลักษณะคล้ายกัน โดยจะอยู่ในรูปของคำสั่งใหม่ที่เราสร้างขึ้นใช้เอง ซึ่งจะอยู่ภายในเครื่องหมายวงเล็บปีกกา เราจะถือว่าคำสั่งที่อยู่ภายในวงเล็บปีกกานั้นเป็นตัวโพรซีเดอร์เสมอ และชื่อโพรซีเดอร์นั้นจะถูกเก็บไว้ยัง dictionary stack ดังที่ได้กล่าวมาแล้วในเรื่องสแต็กข้างต้น จากตัวอย่างที่ 3 จะเห็นว่าโพรซีเดอร์ของ repeat loop จะใช้ bind operator เพิ่มเข้ามา เพื่อเป็นการบอกว่าโพรซีเดอร์นั้นจะถูกแทนด้วยโค้ดที่แท้จริง ซึ่งมักจะทำให้การทำงานของโพรซีเดอร์นั้นเร็วขึ้น อีกทั้งการแก้ไขภายหลังจะไม่ทำให้การทำงานของโพรซีเดอร์นั้นเปลี่ยนแปลงไป

บทส่งท้าย

เมื่อมาถึง ณ จุดนี้แล้ว จากการที่ท่านผู้อ่านได้อ่านเรื่องราวเกี่ยวกับภาษาโพสต์สคริปต์ตั้งแต่ประวัติความเป็นมา คุณสมบัติต่าง ๆ ของภาษา พร้อมทั้งตัวอย่างของโปรแกรมง่าย ๆ เพื่อประกอบคำอธิบายสิ่งสำคัญต่าง ๆ ที่น่าสนใจ ผู้เขียนหวังว่าผู้อ่านหลายท่านคงได้มุมมองของภาษาโพสต์สคริปต์ที่อาจจะสามารถนำไปประยุกต์ใช้ในงานกราฟิกต่าง ๆ

ถึงแม้ว่าภาษาโพสต์สคริปต์นั้นเป็นภาษาที่อาจจะยังไม่เป็นที่นิยมหรือเป็นที่รู้จักกันนักของโปรแกรมเมอร์เท่าที่ควร แต่มันก็นับว่าเป็น Page Description Language ที่ดีตัวหนึ่งทีเดียว และการที่เราได้เขียนโปรแกรมภาษาใหม่ ๆ ที่ยังไม่ค่อยมีใครจะได้ลองศึกษานั้น ผู้เขียนคิดว่าเป็นสิ่งที่น่าท้าทายและน่าสนใจศึกษาเป็นอย่างยิ่ง ในฉบับต่อไปผู้เขียนจะได้นำเสนอเรื่องราวเกี่ยวกับภาษาโพสต์สคริปต์ในระดับสูงต่อไป และจะได้เพิ่มตัวอย่างประกอบคำอธิบายเพื่อให้ท่านผู้อ่านได้เข้าใจรายละเอียดมากยิ่งขึ้นด้วย